UNLIMITED

(2)

AD-A200 644

RSRE MEMORANDUM No. 4157

ROYAL SIGNALS & RADAR ESTABLISHMENT

A COMPARISON OF NONLINEAR OPTIMISATION STRATEGIES FOR FEED-FORWARD ADAPTIVE LAYERED NETWORKS

Author: A R Webb, D Lowe, M D Bedworth

PROCUREMENT EXECUTIVE,
MINISTRY OF DEFENCE,
R S R E MALVERN,
WORCS.

RSRE MEMORANDUM No. 4157

UNLIMITED 88 1024 [illegible]

Royal Signals and Radar Establishment

Memorandum 4157

A Comparison of Nonlinear Optimisation Strategies for Feed–Forward Adaptive Layered Networks. [a]

A.R. Webb, David Lowe, M.D. Bedworth
Speech Research Unit, SP4

5th July 1988

Abstract

This work discusses various learning strategies which may be employed for the generic class of layered feed–forward adaptive networks exemplified by the traditional *Multilayer Perceptron*. Such a network is only useful if a set of weight values exists which allows the network to form a good approximation to an underlying (and possibly unknown) transformation between input and output patterns. This memorandum is motivated by the need for schemes which are capable of producing such a set of weights, (should one exist) as efficiently as possible. As these issues are dependent on specific problem features, this memorandum considers the application of various 'learning' algorithms to examples ranging from small scale 'trivial' problems (solution of the XOR function), to larger scale pattern processing applications (speech recognition of isolated confusable whole words).

[a]Work performed at RSRE between November 1987 and May 1988

Contents

1 Introduction. **1**

2 The 'standard' feed forward adaptive network. **2**

3 Nonlinear optimisation strategies. **4**

3.1 Steepest descent 5

3.2 Conjugate gradients. 5

3.2.1 Accelerated steepest descent and conjugate gradients. 8

3.3 Quasi-Newton, or Variable Metric methods. 8

3.4 The Levenberg-Marquardt approach. 11

4 'Training' the network. **13**

4.1 'Testing' the network. 15

5 Applications and results. **15**

5.1 The Markov Source Discriminator. 16

5.2 *The XOR function* *19*

5.3 Chaotic time series prediction. 21

5.4 Point Source Location Using Focal Plane Arrays. 23

5.5 Speech recognition:
the 'EE' set. 26

6 Discussion. **28**

1 Introduction.

In complex pattern recognition problems by machines, the identification and extraction of relevant information for a specific task, for instance classification, has always been a central issue. In certain classes of problem such as object extraction from natural images, the relevant information may involve edge detection, shape and 'texture' analysis and other *a priori* concepts which help to parameterise the object within a specific environment. In such instances, this *a priori* knowledge may be exploited to construct **explicit** models of the object under scrutiny. Thus, provided with knowledge of the input data ***given*** information about the model, one may extract information of the existence of the model within the input data. This is a situation where explicit model building can be a very efficient strategy.

There are alternative pattern analysis problems, for instance speech recognition, where it is not evident what are the best underlying suitable 'primitives' or relevant features which need to be identified to allow an appropriate decision to be made. Of course, models may be constructed for concepts which we may consider to be relevant (such as the quasi-stationarity of acoustic features, e.g formants, to aid in automatic speech recognition by machine), and the parameters of the model adjusted to represent the best possible description of input data given that particular model. If the explicitly constructed model is an adequate representation of the underlying structure in the input data, then good performance may be expected. Actually, how well a model performs is only valid with hindsight since its utility is only apparent when applied to novel input data — information which has not been previously employed to adjust the parameters of the model. Only if the model has captured the essence of the underlying structure in the data, rather than simply fitting the input data with as many adjustable parameters as desired, will such an explicit model-based scheme succeed well when presented with slightly different information.

Adaptive layered network structures applied to the analysis of complex pattern processing tasks have attracted attention because they appear to have the ability to construct internal representations of important features in input data without having to build an explicit model to describe the source which generated the data [1]. However, this *in principle* capability would be of little use without a well defined strategy, or 'learning rule', which allowed the network to adapt its parameters in such a way as to provide an internal model of the patterns in the input data.

In this paper, we examine various learning rules applied to an adaptive feed-forward layered network model. The motivations for this study are practical ones. A learning rule should satisfy at least two criteria: it should be capable of producing the 'correct' answer, 'most of the time', and in addition it should produce its answer 'within a reasonable time scale'. These are all subjective criteria of course, but in practice it is usually evident when a correct answer is obtained (even getting stuck in a local minimum is not detrimental if that minimum produces a solution which is close enough to the desired answer). The criteria ***most of the time*** and ***within a reasonable time scale*** are even more subjective since they are interrelated. An algorithm which is very fast but does not get a high percentage of 'correct' answers from an initial random configuration may be preferable to a slower technique which is more robust to initial conditions since a larger set of initial conditions may be explored for a given amount of computing time. This is a point worth emphasising. Even for a fixed

[1]Ignoring for the moment the fact that the definition of the geometry and connectivity of an adaptive network constitutes an explict, although flexible, model itself

geometry adaptive layered network structure, the state it will evolve to for fixed initial conditions will be dependent on the learning rule used to drive the evolution. In the course of this paper, there are thus two themes we wish to explore; for a given network geometry and for specific problems, how do the various learning rules perform in terms of computing time and what is the accuracy of the final state compared with the desired targets. This should allow us to infer what is the most efficient adaptive layered network system to employ for the considered class of problems. As the efficiency is strongly dependent on the type of pattern processing u[illegible] considered, and in particular, how the techniques scale with the size and complexity of the problem, we will study a set of test problems finally directed towards *automatic speech recognition* by machine.

We begin by describing our archetypal feed forward layered network structure. Then a brief discussion of four nonlinear optimisation strategies is introduced. The marriage of these learning rules with the layered network structure is then described and subsequently applied to a set of test problems in order to obtain performance estimates. Unless otherwise stated, all simulations were performed on a microvax II running FORTRAN programs under VAX/VMS. No special library routines were employed and the source code was not optimised for performance. Therefore, all timing information given is provided to convey merely an impression of the order of magnitude of the relative computing times involved.

2 The 'standard' feed forward adaptive network.

The structure of the standard layered network model considered in this paper is depicted in Figure 1. It is envisaged that input data may be represented by an arbitrary (real valued) n-dimensional vector, $|x\rangle$[2], or an ordered sequence of n real valued numbers, $\{x_i;\ i = 1, \ldots, n\}$. Thus there are n independent input nodes to the network which accept each input data vector. Each input node is totally connected to a set of n_0 'hidden' nodes (hidden from direct interaction with the environment). Associated with each link between the i-th input node and the j-th hidden node is a scalar μ_{ij}. Usually, the fan-in to a hidden node takes the form of a hyperplane: the input to node j is of the form $\theta_j = \sum_{i=1}^{n} x_i \mu_{ij} = \langle x|\mu_j\rangle$ where $|\mu_j\rangle$ is the vector of n scalar values associated with hidden node j. The rôle of each hidden node is to accept the value provided by the fan-in and output a value obtained by passing it through a (generally, though not necessarily) *nonlinear transfer function*,

$$\phi_j = \phi(\mu_{0j} + \theta_j) = \phi(\mu_{0j} + \langle x|\mu_j\rangle) \tag{1}$$

where μ_{0j} is a local 'bias' associated with each hidden node.

The hidden layer is fully connected to a set of n' output nodes corresponding to the components of an n' dimensional output space. The strength of the connection from the j-th hidden node to the k-th output node is denoted λ_{jk} and thus the value received at the k-th output node is *a weighted sum of the output values from all of the hidden nodes*, $\sum_{j=1}^{n_0} \lambda_{jk}\phi_j$.

In general, the output from the k-th output node will be a nonlinear function of its input, $O_k = \Phi_k(\lambda_{0k} + \langle\lambda_k|\phi\rangle)$ where λ_{0k} is a 'bias' associated with that output node.

[2]This paper employs the bra-ket notation for vectors. A column vector $(x_1, x_2, \ldots)$ is written as $|x\rangle$ (the 'ket'). The corresponding row vector is denoted $\langle x|$ (the 'bra' vector). A scalar product between $|x\rangle$ and $\langle y|$ is given by $\langle y|x\rangle$ and $|y\rangle\langle x|$ is a linear operator with matrix elemente $A_{ij} = y_i x_j$.

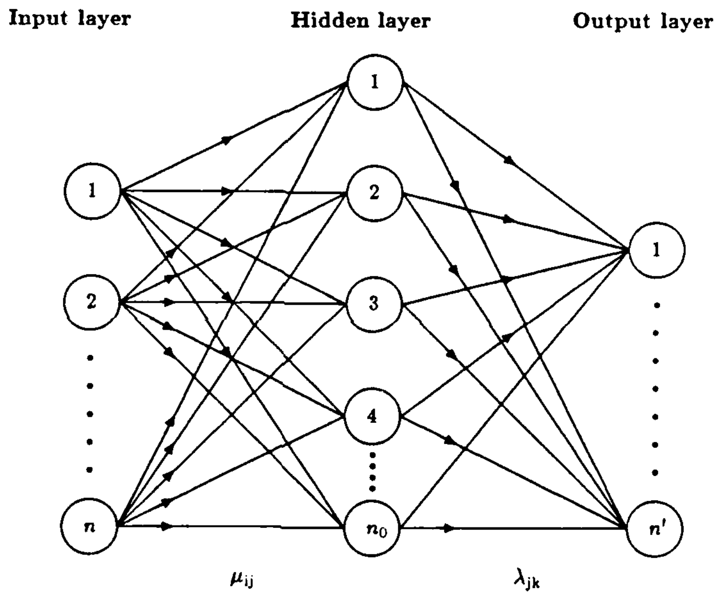

Figure 1: A schematic diagram of the standard feed forward adaptive layered network geometry considered in this paper.

Thus this network provides a transformation mapping from an n–dimensional input space to an n'–dimensional output space via an intermediate characterisation space. This mapping is totally defined by the topology of the network (in particular, how many hidden units are employed) once all the nonlinear transfer functions are specified and the set of weights and biases $\{\lambda, \mu\}$ have been determined. Note that because of the general nonlinearities and the potential reduction of dimensionality, the network performs a nonlinear, noninvertible transformation of the input data. Questions about whether the total possible class of nonlinear transformations which the network is capable of performing is sufficient to encompass the relevant structure in presented input data may only be addressed *a posteriori*. However, even if this class of transformations is sufficient, it may be that the desired solution is not practically obtainable since the choice of weight values which would produce the appropriate transformation is not attainable.

We have yet to indicate how this network can adapt its weight values in order to capture hidden correlations in presented input data. The next section provides such a set of 'learning rules'.

3 Nonlinear optimisation strategies.

This section concerns the general problem of maximisation or minimisation of a function, S, in a high dimensional space. An extremum may be global or local (the extremum being located in a finite region) and except for linear problems, very little is known about finding global extrema for arbitrary nonlinear problems. This section is concerned with local minima.

To evaluate a local minimum, a common strategy is to start from many random initial positions and evolve along a surface in the high dimensional space according to an evolution prescription until an extremum is found. By choosing the most extreme of all the minima, found, one hopes that a good solution to the minimisation has been obtained. In this section we consider four such evolution prescriptions under the assumption that gradient information of the function to be minimised is readily available. The basic idea behind all the strategies is that the evolution proceeds iteratively such that a new vector position at time $t+1$, $|x^{t+1}\rangle$ is chosen in terms of the position at time t, $|x^t\rangle$ by

$$|x^{t+1}\rangle = |x^t\rangle - k\mathbf{B}|g\rangle \tag{2}$$

where $|g\rangle$ is the vector gradient evaluated at $|x^t\rangle$ with components $g_j = \partial S(|x\rangle)/\partial x_j$, $\mathbf{B}$ is a transformation matrix of the gradient producing a search direction and k is a step length.

It is easy to see where Equation (2) stems from. At a local minimum in the high dimensional space we have

$$|g(|x^{t+1}\rangle)\rangle = 0 \tag{3}$$

The lowest order expansion about $|x^t\rangle$ for the gradient gives

$$|g(|x^{t+1}\rangle)\rangle = |g(|x^t\rangle)\rangle + \mathbf{H}(|x^t\rangle)(|x^{t+1}\rangle - |x^t\rangle) \tag{4}$$

where $\mathbf{H}(|x\rangle)$ is is the Hessian matrix with components

$$\begin{aligned} H_{ij} &= \partial g_i(|x\rangle)/\partial x_j \\ &= \partial^2 S(|x\rangle)/\partial x_i \partial x_j. \end{aligned} \tag{5}$$

Assuming that **H** has an inverse, then at the minimum we have

$$|x^{t+1}\rangle = |x^t\rangle - \mathbf{H}^{-1}(|x^t\rangle)|g(|x^t\rangle)\rangle \tag{6}$$

This is the same expression as Equation (2) with $k = 1$ and $\mathbf{B} = \mathbf{H}^{-1}$.

The four strategies below indicate methods for choosing k and **B** (further details and references may be obtained from [1,2,3]).

3.1 Steepest descent

This embodies the simplest nontrivial choice. In this case the transformation matrix **B** is just the n-dimensional identity matrix $\mathbf{1}_n$ and the step length is any fixed value such that $S(|x^{t+1}\rangle) < S(|x^t\rangle)$. This leads to the steepest descent rule:

> *Start at a point,* $|x^0\rangle$. *As many times as necessary, move from the current point* $|x^t\rangle$ in the direction of the local downhill gradient, by the amount $-k\nabla S(|x^t\rangle)$ where k is a fixed constant of proportionality.

The problem with this scheme is that when a minimum is reached in a given single direction in the search space, the update for a new direction will always be perpendicular to the old direction [3]. This will have the undesirable tendency to cause the evolution path to zig-zag even though the true path to follow may be a smooth, gently curving valley on the error surface.

To some extent, this oscillatory behaviour may be damped by including a 'momentum' term into the evaluation of the new direction, thus

$$\begin{aligned} |x^{t+1}\rangle &= |x^t\rangle - k|g^t\rangle + m(|x^t\rangle - |x^{t-1}\rangle) \\ &= |x^t\rangle + k|h^t\rangle \\ |h^t\rangle &= -|g^t\rangle + \gamma\{|x^t\rangle - |x^{t-1}\rangle\} \end{aligned} \tag{7}$$

where $|h^t\rangle$ is the search direction. There are now two parameters to choose: the step length k (or 'learning' rate) and the 'momentum' term $\gamma = m/k$. We will show in the next section that the conjugate gradients algorithm introduces a scheme to adjust these parameters automatically and dynamically.

3.2 Conjugate gradients.

We can see that the problem with steepest descents is that the set of search directions generated is not linearly independent. It would be more efficient to explore the search space using a linearly independent set of search directions which were 'conjugate' to each

[3]since, at a minimum $|g^t\rangle = 0 \Rightarrow \langle g^t|g^{t+1}\rangle = 0$ and hence the new search direction is orthogonal to the previous direction.

other with respect to some positive definite matrix **A** (such that $\langle h^i|\mathbf{A}|h^j\rangle = 0$, $i \neq j$ for each search direction). More specifically, consider the general quadratic form

$$S = S_0 - \langle a|x\rangle + \frac{1}{2}\langle x|\mathbf{H}|x\rangle \tag{8}$$

obtained by expanding the function to be minimised about the minimum value. The gradient of the general quadratic form is

$$|g\rangle = \mathbf{H}|x\rangle - |a\rangle \tag{9}$$

Given an initial search direction $|q^0\rangle$ which is 'downhill' (so that $\langle q^0|g^0\rangle < 0$) the problem is to construct a set of search directions $\{|h^t\rangle\}$ conjugate with respect to the Hessian

$$\langle h^j|\mathbf{H}|h^i\rangle = 0 \quad j > i \tag{10}$$

This may be achieved as follows. At time step t, an arbitrary 'first guess' search direction $|q^t\rangle$ is chosen. Then construct a better search direction

$$|h^t\rangle = |q^t\rangle + \sum_{j=1}^{t-1} z_{tj}|h^j\rangle \tag{11}$$

and choose the coefficients z_{ij} so that Equation (10) is satisfied. To do this, apply $\langle h^k|\mathbf{H}$ to both sides of (11) and exploit (10) to give

$$z_{tk} = -\frac{\langle h^k|\mathbf{H}|q^t\rangle}{\langle h^k|\mathbf{H}|h^k\rangle} \tag{12}$$

Note that the denominator above is non-zero if **H** is positive definite and *the search direction* is not the null vector. This expression still depends on the particular choice of a 'random' search direction. For the usual and specific case of the local downhill gradient $|q^t\rangle = -|g^t\rangle$ we have

$$z_{tj} = \frac{\langle h^j|\mathbf{H}|g^t\rangle}{\langle h^j|\mathbf{H}|h^j\rangle} \tag{13}$$

Unfortunately, this requires the evaluation of the Hessian matrix which is not generally available. However, for quadratic forms this is not necessary. Note from Equation (9) that the search directions and the local gradients at times t and $t+1$ are linked by

$$|g^{t+1}\rangle - |g^t\rangle = \lambda_t\mathbf{H}|h^t\rangle \tag{14}$$

where $\lambda_t|h^t\rangle$ is a step length taken in the direction of $|h^t\rangle$. Now at a minimum of the function in the direction of the current search direction, we must have

$$\langle h^t|g^{t+1}\rangle = 0 = \langle h^t|g^t\rangle + \lambda_t\langle h^t|\mathbf{H}|h^t\rangle \tag{15}$$

which implies that the step length is just

$$\lambda_t = -\frac{\langle h^t|g^t\rangle}{\langle h^t|\mathbf{H}|h^t\rangle} \tag{16}$$

In addition, at a minimum the local gradient is orthogonal to all the previous search directions, and successive local gradients are orthogonal for a quadratic form.

We may exploit (14) and substitute for the Hessian in Equation (13) to give the coefficients in terms of successive local gradients and search directions alone:

$$\begin{aligned} z_{tj} &= \frac{\langle g^t|g^{j+1}\rangle - \langle g^t|g^j\rangle}{\langle h^j|g^{j+1}\rangle - \langle h^j|g^j\rangle} \\ &= \frac{\langle g^t|g^{j+1}\rangle - \langle g^t|g^j\rangle}{\langle g^j|g^j\rangle - \langle g^j|g^{j+1}\rangle + \sum_{k=1}^{j-1} z_{jk}[\langle h^k|g^{j+1}\rangle - \langle h^k|g^j\rangle]} \end{aligned} \tag{17}$$

However, for a true quadratic form we know that $\langle g^t|g^j\rangle = 0$ for $j < t$ and also $\langle h^k|g^j\rangle = 0$ for $k < j$. Therefore, strictly speaking this gives an expansion for the coefficients which minimises the quadratic form as

$$z_{t,t-1} = \frac{||g^t||^2}{||g^{t-1}||^2} \tag{18}$$

This is known as the Fletcher-Reeves choice [2]. Of course for non-quadratic forms the arguments presented do not strictly apply. The general strategy is to ignore this fact and proceed in the knowledge that near a local minimum, the equations should represent a good approximation to the actual surface for a sufficiently smooth search space. However, once we relax the strict quadratic assumptions, and in particular no longer insist that successive local gradients are orthogonal (so that $\langle g^t|g^{t+1}\rangle \neq 0$), this leads to a higher order approximation to the coefficients known as the Polak-Ribiere choice [2]:

$$z_{t,t-1} = \frac{||g^t||^2 - \langle g^t|g^{t-1}\rangle}{||g^{t-1}||^2} \tag{19}$$

This is the particular choice which most researchers use in practice as it tends to be more robust than the Fletcher-Reeves choice.

Actually, one sees from Equation (17) that a better higher order approximation to use would be

$$z_{t,t-1} = \frac{||g^t||^2 - \langle g^t|g^{t-1}\rangle}{||g^{t-1}||^2 - \langle g^t|g^{t-1}\rangle} \tag{20}$$

which is a strategy that we have not seen in the literature. We will be comparing these three update strategies later, and so for ease of reference, we will refer to this latter update rule as Update3.

Whichever variant is employed is largely a matter of choice, *a priori* knowledge and hindsight since the theorems used to derive the expressions do not strictly apply beyond the validity of the quadratic form assumption.

In summary then, the conjugate gradients approach to the minimisation of a function proceeds as follows:

Choose an initial start point, $|x^0\rangle$ which determines an initial direction, $|h^0\rangle = -|g(|x^0\rangle)\rangle$. For each subsequent time step choose a new search direction, $|h^t\rangle$ according to

$$|h^t\rangle = -|g^t\rangle + z_{t,t-1}|h^{t-1}\rangle$$

where $|g^t\rangle = |g(|x^t\rangle)\rangle$ is the local gradient at time t and $|x^t\rangle = |x^{t-1}\rangle + \lambda_{t-1}|h^{t-1}\rangle$ with λ_{t-1} being determined by a linear search to obtain the minimum of the function $S(|x\rangle + \lambda|h\rangle)$.

3.2.1 Accelerated steepest descent and conjugate gradients.

It was noted that that the solution of a nonlinear optimisation scheme based on steepest descents would depend on the choices of 'learning rate' and 'momentum term'. However, by comparing the conjugate gradients and steepest descents strategies, the conjugate gradients approach suggests a dynamic choice for the learning rate and momentum term depending on information at times t and $t+1$. In particular, the momentum term is suggested to be $z_{t,t-1}$ (for instance a momentum term which varies with the squared ratio of successive local gradients, Equation (18)) and the step length chosen to find a minimum in the direction determined by the current search direction.

3.3 Quasi-Newton, or Variable Metric methods.

This subsection considers 'variable metric' or 'quasi-Newton' methods. As in the conjugate gradients case, this class of methods aims to accumulate information from accurate line minimisations such that at most, n such minimisations lead to an exact minimum of a quadratic form in n dimensions. This is performed by generating a sequence of search directions which is conjugate with respect to the Hessian of the function to be minimised. The difference between this approach and conjugate gradients is in the way that the information is stored and updated.

For a quadratic form, from Equation (9) the gradients and positions at step $t+1$ are connected with those at step t by

$$|x^{t+1}\rangle = |x^t\rangle + \mathbf{H}^{-1}(|g^{t+1}\rangle - |g^t\rangle) \tag{21}$$

which is the quasi-Newton condition. Ideally, the new search direction, $|h^t\rangle = |x^{t+1}\rangle - |x^t\rangle$ is obtained with knowledge of the inverse Hessian, However, this is in practice very expensive to obtain and hence the basic idea behind all quasi-Newton methods is to build up a sequence of matrices which tend in the limit to the inverse Hessian (which for an n dimensional quadratic form means that the sequence of approximate matrices will equal the exact inverse Hessian after at most n iterations). Thus the update rule considered in this subsection is of the general form

$$|x^{t+1}\rangle = |x^t\rangle - \lambda_t \mathbf{A}^t |g^t\rangle \tag{22}$$

where $lim_{t\to\infty}\mathbf{A}^t = \mathbf{H}^{-1}$ and λ_t is a scalar parameter chosen to either minimise, or reduce the function along the search direction $|h^t\rangle$.

Thus the problem is to devise an iterative scheme for updating the transformation matrix at time t to produce a new transformation matrix at time $t+1$. In addition, perhaps we would like to impose the constraint that successive approximation matrices also should have to satisy the quasi-Newton condition

$$\begin{aligned} |h^t\rangle &\equiv |x^{t+1}\rangle - |x^t\rangle \\ &= \mathbf{A}^{t+1}[|g^{t+1}\rangle - |g^t\rangle \\ &\equiv \mathbf{A}^{t+1}|y^t\rangle \end{aligned} \tag{23}$$

One such class of iteration schemes is known as the Huang class [4]

$$\mathbf{A}^{t+1} = \mathbf{A}^t - \mathbf{A}^t|y^t\rangle\langle q^t| + \rho_t|s^t\rangle\langle p^t| \tag{24}$$

where ρ_t is an arbitrary scalar parameter and the vectors $|q^t\rangle$ and $|p^t\rangle$ are defined as

$$\begin{aligned}|p^t\rangle &= \mu_t|s^t\rangle + \nu_t(\mathbf{A}^t)^T|y^t\rangle \\ |q^t\rangle &= \xi_t|s^t\rangle + \chi_t(\mathbf{A}^t)^T|y^t\rangle\end{aligned} \tag{25}$$

and are subject to the restriction

$$\langle p^t|y^t\rangle = \langle q^t|y^t\rangle = 1$$

These latter two conditions impose restrictions on the possible values of μ, ν, ξ, χ and thus the Huang class is a three parameter family of update rules.

Note that the Huang class obeys the relation

$$\mathbf{A}^{t+1}|y^t\rangle = \rho_t|s^t\rangle \tag{26}$$

and so if we insist that the quasi-Newton condition is obeyed (which implies that $\rho_t \equiv 1$) then the general update family is a two parameter class. The Huang family contains all of the popular variable metric update strategies, but the primary importance of the class is contained in a list of properties which all members of the family share. In particular

- if the line minimisation is performed exactly, then for a quadratic form the algorithm terminates with the exact solution in at most n steps
- $$\mathbf{A}^k|y^t\rangle = \rho_t|s^t\rangle \quad t < k$$
- $$\langle s^{t'}|\mathbf{H}|s^t\rangle = 0 \quad t \neq t'$$
- if the process requires n steps to terminate and $\rho_t = 1$ then $\mathbf{A}^n \equiv \mathbf{H}^{-1}$

A final very interesting property proved by Dixon [5] is that if the line search is exact, then all members of the Huang class with the same ρ_t at each step generate an identical sequence of search directions for continuous functions. Since experiments using different members of the Huang class give rise to different performance — and in particular, to widely differing sequences $\{|x^t\rangle\}$ — then one can conclude that either the process is very susceptible to numerical error, or the line searches employed have not been perfect and this point is a crucial factor. The question of the relative merits of exact but expensive line searches allowing quadratic termination versus approximate line searches is an issue which we cannot address in this memorandum. We will, however, comment on our own experiences after the numerical simulations.

Note that the Huang family produces a sequence of matrices, $\{\mathbf{A}^{t+1}\}$ which need not be symmetric, in contrast to the Hessian which it eventually approximates. Choosing $\rho_t = 1$ and the parameters μ, ν, ξ, χ so as to preserve the symmetry of $\mathbf{A}^{t+1}$, leads to a subclass of the Huang family known as the single parameter Broyden family [6]. Specifically, the choice

$$\begin{aligned}\rho_t &= 1\\ \mu_t &= \eta_t \frac{\langle y^t|\mathbf{A}^t|y^t\rangle}{\langle s^t|y^t\rangle}\\ \chi_t &= -\mu_t\\ \nu_t &= -\eta_t + \frac{1}{\langle y^t|\mathbf{A}^t|y^t\rangle}\\ \xi_t &= \left|\frac{\langle y^t|\mathbf{A}^t|y^t\rangle}{\langle s^t|y^t\rangle}\right|^2 \left\{\eta_t + \frac{\langle s^t|y^t\rangle}{\langle y^t|\mathbf{A}^t|y^t\rangle^2}\right\}\end{aligned}$$

gives the Broyden family

$$\begin{aligned}\mathbf{A}^{t+1} = \mathbf{A}^t &- \frac{\mathbf{A}^t|y^t\rangle\langle y^t|\mathbf{A}^t}{\langle y^t|\mathbf{A}^t|y^t\rangle}\\ &+ \frac{|s^t\rangle\langle s^t|}{\langle s^t|y^t\rangle} + \eta_t|v^t\rangle\langle v^t|\end{aligned} \tag{27}$$

where

$$|v^t\rangle = \mathbf{A}^t|y^t\rangle - |s^t\rangle\frac{\langle y^t|\mathbf{A}^t|y^t\rangle}{\langle s^t|y^t\rangle} \tag{28}$$

and η_t is an arbitrary scalar parameter.

The Broyden family incorporates the two most popular variable metric methods; the Davidon–Fletcher–Powell (DFP) technique (obtained by setting $\eta_t = 0$) and the Broyden–Fletcher–Goldfarb–Shanno (BFGS) update (obtained by setting $\eta_t = 1/\langle y^t|\mathbf{A}^t|y^t\rangle$). Most researchers in the field of nonlinear optimisation tend to agree that, in practice, the BFGS update is more robust than the DFP approach — a fact which is very probably associated with the inexact line searches. To check this opinion, we have employed both the DFP and BFGS update rules as representative of the general class of quasi-Newton methods. We have no experience of whether these methods are suitable for the class of problems we have considered, and it may well be that a different choice of η_t could work more efficiently in practical problems associated with speech recognition for instance. However, it is possible to pinpoint immediately one problem with the general class of variable metric methods — the necessity to store and update a matrix of size $N \times N$. For a speech recognition problem where a speech pattern is treated as static and provided over a time interval of a second, this value of N can be typically of the order of 6000 (see subsection 5.5).

Thus the 'learning rule' appropriate to variable metric or quasi-Newton strategies may be posed as follows:

From each current position, $|x^t\rangle$, the new position is determined by

$$|x^t\rangle = |x^t\rangle + \lambda_t|h^t\rangle$$

where λ_t is chosen to minimise the function $S(|x\rangle + \lambda|h\rangle)$ in the direction of $|h\rangle$. The new direction, $|h^{t+1}\rangle$ is given by

$$|h^t\rangle = -\mathbf{A}^t|g^t\rangle$$

where the transformation matrix is given by Equation (27).

3.4 The Levenberg–Marquardt approach.

The Levenberg-Marquardt method is a technique for minimising a nonlinear sum-of-squares function of the form

$$S(|x\rangle) = \langle f|f\rangle \tag{29}$$

where at least one of the $f_i(|x\rangle), i = 1, \ldots, M$ depends nonlinearly on $|x\rangle$. The minimisation of $S(|x\rangle)$ could be approached by an algorithm for the minimisation of a general function of N variables like those described above, but the Levenberg-Marquardt method takes explicit note of the sum-of-squares form of $S(|x\rangle)$.

Levenberg's approach [7] was to consider the Gauss-Newton method (or Taylor series method) but with an additional damping term in order to ensure that the estimate of the position of the minimum does correspond to a decrease in S. More specifically, consider the expansion of $|f\rangle$ about a point $|x^0\rangle$ (at which it is assumed that S does not have a stationary value). To first order we have

$$|f(|x\rangle)\rangle \approx |F(|x\rangle)\rangle = |f(|x^0\rangle)\rangle + \mathbf{J}^T(|x\rangle - |x^0\rangle), \tag{30}$$

where $\mathbf{J}$ is the Jacobian matrix evaluated at $|x^0\rangle$

$$\mathbf{J}_{ij} = \frac{\partial f_i(|x\rangle)}{\partial x_j}. \tag{31}$$

The Gauss-Newton method consists of minimising an approximation, Φ, to S, given by

$$\Phi(|x\rangle) = \langle F|F\rangle \tag{32}$$

Differentiating this expression with respect to the parameters $|x\rangle$ and equating to zero gives the usual 'normal equation' of the linear least-squares problem,

$$\mathbf{J}^T\mathbf{J}(|x\rangle - |x^0\rangle) + \mathbf{J}^T|f\rangle = 0 \tag{33}$$

with solution for $|x\rangle$

$$|x\rangle = |x^0\rangle - (\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T|f\rangle \tag{34}$$

provided the inverse $(\mathbf{J}^T\mathbf{J})^{-1}$ exists.

Noting that the vector, $\mathbf{J}^T|f\rangle$ is simply the gradient of the function Φ at the point $|x^0\rangle$, then Equation (34) is identical in form to Equation (6) with the Hessian matrix approximated by the matrix product $\mathbf{J}^T\mathbf{J}$. The full expression for the Hessian matrix, $\mathbf{H}$, is

$$\mathbf{H} = \mathbf{J}^T\mathbf{J} + \mathbf{K}, \tag{35}$$

where the matrix $\mathbf{K}$ involves second derivative terms and is given by

$$\mathbf{K}_{ij} = \sum_{k=1}^{M} f_k \frac{\partial^2 f_k}{\partial x_i \partial x_j}. \tag{36}$$

The second derivative term may be dismissed when it is small enough to be neglected compared to the term involving the first derivatives, or if the function values are small. Thus, the Gauss-Newton method (Equation (34)) is an approximate inverse Hessian method.

One of the problems with the solution for $|x\rangle$ obtained from Equation (34) is that the values of the increments obtained may be so large as to invalidate the approximation given by Equation (30), so that a decrease in Φ may not correspond to a decrease in S. One solution to this problem is to introduce a step size k, $(0 < k \leq 1)$ into Equation (34) and to search along the direction $-(\mathbf{J}^T\mathbf{J})^{-1}\mathbf{J}^T|f\rangle$ to find the minimum of S. Levenberg's proposal [7] is to limit the size of the increments, $|x\rangle - |x^0\rangle$, in order to improve the first-order approximation given by Equation (30) and to minimise simultaneously the quantity Φ. The increments are minimised in a least-squares sense and so the total expression to be minimised is

$$\tilde{\Phi} = w\Phi + (\langle x| - \langle x^0|)\mathbf{B}(|x\rangle - |x^0\rangle), \tag{37}$$

where $\mathbf{B}$ is a diagonal matrix with positive elements which express the relative importance of damping different elements of the increment vector, and w is a positive quantity expressing the relative importance of the residuals and the increments in the minimisation. Differentiating $\tilde{\Phi}$ with respect to the parameters $|x\rangle$ and equating to zero gives the solution for $|x\rangle$

$$|x\rangle = |x^0\rangle - (\mathbf{J}^T\mathbf{J} + \frac{1}{w}\mathbf{B})^{-1}\mathbf{J}^T|f\rangle \tag{38}$$

The standard Gauss-Newton least-squares method is a special case with $w \to \infty$, and it can be shown that there is a value for w for which the sum of squares of the true residuals (Equation (29)) is reduced [7]. If the matrix $\mathbf{B}$ is taken to be the identity matrix Equation (38) becomes

$$|x\rangle = |x^0\rangle - (\mathbf{J}^T\mathbf{J} + \epsilon\mathbf{I})^{-1}\mathbf{J}^T|f\rangle. \tag{39}$$

where $\epsilon = \frac{1}{w}$. As $\epsilon \to \infty$, the direction of the increment tends to the steepest descent direction.

Levenberg [7] and Marquardt [8] introduce scaled parameters in order to improve numerical aspects of the computation. Defining $|x\rangle^* = \mathbf{D}|x\rangle$, where $\mathbf{D}$ is a diagonal matrix having positive diagonal elements, then the tranformed Jacobian is $\mathbf{J}^* = \mathbf{J}\mathbf{D}^{-1}$. Introducing the scaled parameters into Equation (39) gives an equation for the unscaled parameters

$$|x\rangle = |x^0\rangle - (\mathbf{J}^T\mathbf{J} + \epsilon\mathbf{D}^2)^{-1}\mathbf{J}^T|f\rangle. \tag{40}$$

The particular scaling proposed by Marquardt is

$$\mathbf{D}_{ii} = (\mathbf{J}^T\mathbf{J})_{ii} \tag{41}$$

Thus, the Levenberg-Marquardt method is a scheme for varying smoothly between the steepest descent method ($\epsilon \to \infty$) and the Gauss-Newton method ($\epsilon \to 0$) and may be applied in its scaled (Equation (39)) or unscaled forms (Equation (41)). One parameter to be chosen is the value of ϵ. Marquardt [8] proposes an iterative scheme. Initially, $\epsilon = 0.1$ and it is reduced by a factor of ten if the previous solution in the iteration, $|x^{t-1}\rangle$, satisfies

$$S(|x^t\rangle) < S(|x^{t-1}\rangle) \tag{42}$$

and the values of the parameters are updated. However, if

$$S(|x^t\rangle) \geq S(|x^{t-1}\rangle) \tag{43}$$

then ϵ is increased by a factor of ten and a new solution for $|x^t\rangle$ is sought using Equation (39). As ϵ becomes large, a decrease in S is guaranteed. This scheme has the advantage that the direction and steplength are determined simultaneously. An alternative approach is to adopt a fixed expression for ϵ [9], for example

$$\epsilon = \| \, |f(|x\rangle)\rangle \, \| \tag{44}$$

for any norm and to perform a line search along the direction $-(\mathbf{J}^T\mathbf{J}+\epsilon\mathbf{I})^{-1}\mathbf{J}^T|f\rangle$. This is the approach which we have adopted in the subsequent numerical simulations.

In summary, the Levenberg-Marquardt method gives the following strategy for finding the minimum of the nonlinear least-squares problem.

From each current position, $|x^t\rangle$, a new position is determined by

$$|x^{t+1}\rangle = |x^t\rangle + k_t|x^{t-1}\rangle \tag{45}$$

where k_t is chosen to minimise the sum-of-squares function $S(|x^t\rangle + k_t|h^t\rangle)$ in the direction $|h^t\rangle$ given by

$$|h^t\rangle = -(\mathbf{J}^T\mathbf{J}+\epsilon\mathbf{I})^{-1}\mathbf{J}^{\mathbf{T}}|\mathbf{f}\rangle \tag{46}$$

and $\epsilon = \|\,|f\rangle\,\| = \langle f|f\rangle$ for a Euclidean distance function.

4 'Training' the network.

In section 2, the architecture of the 'standard' feed forward adaptive layered network was introduced. The network will operate once a set of weight values $\{\lambda_{jk},\ \mu_{ij}\}$ has been determined. This set is conditional upon training data presented in the form of representative input and corresponding target output patterns. However, no information was given in that section as to how these weights could be obtained.

Section 3 discussed various approaches to unconstrained minimisation of a nonlinear function under assumptions of continuity and differentiability. This section marries these two concepts together.

The basic utility of a 'trained' adaptive network has to be that when presented with a novel input pattern, the output pattern should be a sensible generalisation from the training patterns. In order to generalise adequately, it should be capable of interpolating from the information provided in the training data. For this to be so, an interpolating surface must be produced which is smooth in some sense and has to be constructed in such a way as to pass as close as possible to all the training data. It may be possible to produce a surface which **exactly** passes through each one of the training data points specified by the set of input-output pairs

$$\{|I^p\rangle,\ |T^p\rangle,\ p=1,2,\ldots P\} \in \mathbf{R}^n \otimes \mathbf{R}^{n'}$$

where $|I^p\rangle \in \mathbf{R}^n$ is the input n-dimensional pattern and $|T^p\rangle \in \mathbf{R}^{n'}$ is the desired n'-dimensional target pattern. However, this may not be necessary or even desirable (as in the case of noise-corrupted data) and indeed it may not even be attainable, since the degree of complexity as specified by the architecture of the network may not be sufficient to interpolate exactly all the training points [4]. Thus, the criterion that the network is trainable at all has to imply that the actual outputs of the network are in some sense as close as possible to the desired target patterns. Therefore it is necessary to introduce an error measure associated with the difference between the netwnrk output pattern and the desired target pattern for each of the fixed input training patterns. There are various error

[4]This point relates to the comments of Minsky and Papert [10] on the assumed ability of the Perceptron (and similar multi-layered networks) to be able to *represent* a relationship as a separate issue to be able to *model* that relationship.

measures one could introduce; however for the purposes of this paper we will only consider the sum-of-squares error:-

$$E = \sum_{p=1}^{P} \| \, |T^p\rangle - |O^p\rangle \, \|^2 \tag{47}$$

where the summation runs over all the patterns in the training set. From section 2, the actual output pattern, $|O^p\rangle$ may be expanded in terms of the network weights, the nonlinear transfer functions and the input pattern, $|I^p\rangle$. For a Euclidean distance function, this gives the explicit error measure:-

$$E = \sum_{p=1}^{P} \sum_{k=1}^{n'} \left\{ T_k^p - \Phi_k\Big(\lambda_{0k} + \sum_{j=1}^{n_0} \phi_j[\mu_{0j} + \sum_{i=1}^{n} I_i^p \mu_{ij}]\lambda_{jk}\Big) \right\}^2 \tag{48}$$

Thus the strategy in the 'training phase' of the layered network is to minimise this total error measure. Since it is an explicit differentiable function of all the known parameters it is an ideal candidate problem for the nonlinear optimistion techniques discussed in section 3.

The problem has so far been cast in a general framework. In particular we have the freedom to choose the particular form of the nonlinear transfer function at each node in the hidden and output layers independently. However, for the purpose of analysing the problems posed in the rest of the paper, the network structure is specialised to the case where the transfer function of each output unit is linear ($\Phi_k(x) = x \; \forall \; k$) and the nonlinearity on each hidden node is taken to be the function $\phi_j(x) = 1/[1 + \exp(-x)] \; \forall \; j$. Of course there are instances when it is desirable for the output units also to have this 'logistic' form. For instance, problems where the output pattern values are known to be strictly bounded or are supposed to encode probabilities directly. Generally, it is sensible to build such *a priori* knowledge directly into the network when the main concern is designing a network to solve a specific problem with known attributes and symmetries. This is not the aim of this study. The choice of linear output units simply makes the explicit evaluation of the partial derivatives with respect to each weight and 'bias' easier to compute. It also emphasises the point that each node may have in principle a different nonlinear transfer function as long as it is differentiable.

Thus the adaptive layered network is 'trained' by searching for a local minimum in the error function. This search is performed by employing one of the various nonlinear optimisation strategies discussed in the previous section, since the function to be minimised and all its partial derivatives are readily available.

Note that for an accurate evaluation of the gradient, it is necessary to present all the training patterns to the network before an update procedure is allowed. This is in contrast to the traditional approach to learning in multi-layer perceptrons where the update in the 'error back propagation' is usually performed after the presentation of a small number of training patterns which only gives an estimate of the local gradient. There are, of course, advantages and disadvantages to this approach particularly for large training sets.

Since the nonlinear search strategy cannot be guaranteed to find a 'good' local minimum, it is necessary to repeat the search procedure for a variety of different random start weights to find the deepest minimum. This is used to build up statistics on the efficiency of the search strategy to find an acceptable local minimum and is how the subsequent results were obtained.

4.1 'Testing' the network.

To determine how good a model any given network produces for a certain pattern transformation problem, it is not sufficient to obtain a solution with minimum error on the finite training set. For instance, a simple lookup table explicitly depicting the one-one mapping in the training set would have zero error but would be totally inadequate when applied to a similar pattern not considered in the domain of the table. A trained network is only useful if it has managed to capture similarities or relationships which were evident in the training set and which may therefore be applied to any additional patterns which conform to the same structure. Only if a network succeeds in modelling the *relationships* between input and output patterns, rather than producing a model of the training data itself (such as a lookup table), is the network capable of **generalisation**.

In order to test a network for its ability to generalise, a 'trained' network has to analyse patterns with known labels which have not previously been employed in the training set. For each of p' new input patterns, $|\tilde{I}^i\rangle$ to such a network, the output patterns that the network actually produced, $s^i = s(|\tilde{I}^i\rangle)$ may be compared with the labels, or patterns, $|\tilde{T}^i\rangle$, that were expected. We again employ a sum-squared difference expression, but because this error depends on the number of testing patterns, we choose to employ a normalised error, $\mathcal{E}$

$$\mathcal{E} = \sqrt{\frac{\sum_{i=1}^{p'} \| s^i - |\tilde{T}^i\rangle \|^2}{\sum_{i=1}^{p'} \| |\tilde{T}^i\rangle - |\bar{T}\rangle \|^2}} \tag{49}$$

where $|\bar{T}\rangle$ is the mean target pattern over all the test patterns[5]. This error measure has the advantage that if it has a value of unity, then one can infer that the network is predicting the output *in the mean* and a value of zero means that it is predicting perfectly. As an empirical guideline, a normalised error in the region of 0.1 generally means that the transformation mapping has been modelled adequately for most classification applications. In the results of the next section, the ability of the chosen networks to generalise is indicated by the normalised error averaged over each set of experiments.

5 Applications and results.

The first task is to restrict some of the possible variability that we discussed in section 2 (approximate vs. accurate line search, update strategy in conjugate gradients, BFGS or DFP as a good variable metric approach, implementation of Levenberg-Marquardt etc.). To do this, we have to adopt an empirical criterion and so the following subsection treats a small scale problem, the six bit Markov source discriminator [11], in detail with most of the possible variations that we consider to be reasonable. On the basis of these results, we will choose to adopt the best combination of strategies for each learning algorithm and apply the 'optimum' search techniques to a small range of network problems for an overall comparison of the various approaches.

All the methods implemented perform some form of line minimisation in which the current search direction is examined in order to find the minimum of the function in that direction. The approximate line search strategy uses a linear search to find three positions

[5]We will encounter problems where the 'test' patterns are the same as the 'training' patterns, in which case the normalised error will be evaluated over the training set.

$a < b < c$ such that the values of the function, S, at these positions satisfy $S(b) < S(a)$ and $S(b) < S(c)$. Parabolic inverse interpolation is then performed to find the position of the minimum of the parabola which interpolates the three points $(a, S(a))$, $(b, S(b))$ and $(c, S(c))$. The position of the minimum of the parabola is used as the new value for the parameters. The accurate line search is also based on a line search with inverse parabolic interpolation, but this is performed iteratively to isolate the minimum to a fractional precision of about *tol* (usually set to the square root of the machine's floating point precision).

To test for convergence, a fractional tolerance, *ftol*, was used. Iteration will stop if

$$2.0 \times |S - Sp| \leq ftol \times (|S| + |Sp| + \epsilon) \tag{50}$$

where S is the current function value; Sp is the value on the previous iteration and ϵ is a small positive constant (taken to be 10^{-10}). In the examples considered, a value of 10^{-5} was used for *ftol*, except where otherwise stated

5.1 The Markov Source Discriminator.

This is a problem [11] which has the merits of being small scale but at the same time embodying features in common with real world application associated with speech modelling. The Markov source discriminator has been used as an intermediate point between small, discontinuous problems and large, continuous, stochastic problems. It is a generalisation of the XOR function and is a simple member of a class which includes Markov random fields and hidden Markov models.

Each input pattern is a short binary first-order Markov chain; i.e. a sequence of N zeros and ones in which the probability of a one in any position depends on the symbol at the previous position. Such a pattern is generated by a symmetric Markov source. A symmetric Markov source is completely specified by a transition probability, $p \triangleq P(x_i \neq x_{i-1})$, of any bit being different from the previous bit.

The network should discriminate between two known Markov sources given segments of their state history.

The probability of any particular sequence of symbols, $x_1 \ldots x_N$, being produced by a source with known transition probability, p, is:

$$P(x_1 \ldots x_N \mid p) = \prod_{i=2}^{N} T_i, \tag{51}$$

$$T_i \triangleq \begin{cases} p & if\ x_i \neq x_{i-1} \\ (1-p) & if\ x_i = x_{i-1} \end{cases} \tag{52}$$

and so

$$P(x_1 \ldots x_N \mid p) = p^T (1-p)^{N-1-T}. \tag{53}$$

where T is the number of transitions.

Results are presented for six-bit patterns generated by two equally likely Markov sources with transition probabilities of $p = 0.7$ and $p = 0.3$ (The classic exclusive-OR problem considered in the next subsection is the limiting case of two bit patterns and transition probabilities of 0.0 and 1.0).

There are several possible output codings we could choose to train the network on. For instance, we could choose a 1-from-2 coding corresponding to whether the input pattern was or was not generated from one of the Markov chains. However, in this experiment, we chose to train on a single real number output coding, $P(p = 0.7 \mid x_1 \ldots x_6)$, the probability that given a sequence $x_1 \ldots x_6$, it was generated by the Markov chain with transition probability 0.7.

Using Bayes' rule

$$P(p = 0.7 \mid x_1 \ldots x_N) = \frac{P(x_1 \ldots x_N \mid p = 0.7)P(p = 0.7)}{P(x_1 \ldots x_N)}. \qquad (54)$$

But since both sources are equally likely,

$$P(p = 0.7) = \frac{1}{2} = P(p = 0.3)$$

then

$$P(x_1 \ldots x_N) = \frac{1}{2}(P(x_1 \ldots x_N \mid p = 0.7) + P(x_1 \ldots x_N \mid p = 0.3))$$

and so

$$P(p = 0.7 \mid x_1 \ldots x_N) = \frac{P(x_1 \ldots x_N \mid p = 0.7)}{P(x_1 \ldots x_N \mid p = 0.7) + P(x_1 \ldots x_N \mid p = 0.3)} \qquad (55)$$

$$P(p = 0.7 | x_1 \ldots x_N) = \frac{0.7^T 0.3^{N-1-T}}{0.7^T 0.3^{N-1-T} + 0.3^T 0.7^{N-1-T}}. \qquad (56)$$

This gives the single real valued target pattern used in our experiments.

Thus the network we considered was $6 - n_0 - 1$ where we arbitrarily chose $n_0 = 6$.

Training consisted of presenting the 64 (2^6) distinct six bit input patterns along with the targets $P(p = 0.7 \mid x_1 \ldots x_6)$ for a given set of initial random start weights chosen arbitrarily in the range $(-1, 1)$. After training, the network was tested by recording the networks response to the same 64 input patterns. This procedure was repeated for one hundred different sets of random start weights, and for each set of start weights, one of fifteen possible search strategies was employed (see Table 1). For each experiment, statistics were gathered on how many solutions were correct, how long each experiment took in terms of CPU time, how efficiently each set of experiments associated with a given search strategy obtained the correct answer (in terms of the number of gradient and function calls per experiment which produced a correct result) and how accurately the output approximated the desired targets (in terms of the average normalised error over all the experiments). For the Markov Source Discriminator problem, the criterion of when the answer produced was 'correct' was arbitrarily decided to be when the normalised error was less than 0.25. This threshhold biases favourably towards the standard steepest descent technique which runs for a specified number of iterations irrespective of the error function and thus tends to have a larger normalised error[6]. For this experiment, the value of the fractional tolerance used to

[6]The error was evaluated on the training set, as this is an instance where there is no test data

	% Correct	Number of iterations	Mean error $\mathcal{E}$	Number of function calls	Mean CPU (seconds)
S.D. (standard)	90	2000	0.2142	0	759
S.D. (exact)	96	1292	0.2051	26580	2957
S.D. (approx.)	95	3088	0.2059	18030	2351
C.G. (F-R & exact)	96	287	0.1902	5550	625
C.G. (F-R & approx.)	92	283	0.2161	1654	217
C.G. (P-R & exact)	96	298	0.1726	5737	647
C.G. (P-R & approx.)	96	272	0.1643	1659	217
C.G. (Update3 & exact)	70	278	0.3452	5322	600
C.G. (Update3 & approx.)	67	279	0.3548	1637	215
q-N (BFGS &exact)	100	246	0.0965	4541	525
q-N (BFGS &approx.)	100	258	0.0919	1163	188
q-N (DFP &exact)	100	215	0.1133	4311	503
q-N (DFP &approx.)	100	187	0.1004	1005	153
L-M (exact)	100	129	0.0676	2551	678
L-M (approx.)	100	140	0.0667	671	524

S.D = steepest descent, C.G = conjugate gradients,
q-N = quasi-Newton, L-M = Levenberg-Marquardt,
'exact' and 'approx.' refer to the accuracy of
performing the line search ('standard' steepest
descents does not perform a line search at all of course).
Standard steepest descent used a step length, $k = 0.01$
and a momentum term, $\gamma = 90$ (see eqn. (7) for notation)

Table 1: Results Table illustrating the performance of various search strategies on the Markov Source Discriminator using a 6–6–1 standard layered network with linear output units.

test for convergence was set to 10^{-4}. Another point worth noting regarding the 'standard' steepest descent is that a choice has to be made regarding a good set of values for the step length and the momentum term (see section 3.1 for notation). In practice, it is important to choose these parameters appropriately which has to be performed empirically. This is an additional overhead in terms of time that ought to be taken into account when comparing these methods.

From the table of results for this problem, Table 1, we may infer the following:

- Steepest descents (all variants) is the worst search strategy to employ. The line search variants produced runs which required almost an order of magnitude more iterations, function calls and CPU time than most of the other techniques. In addition, this excessive computing time was not offset by producing answers with a smaller normalised error. Even truncating the maximum number of iterations allowed produced runs which required more CPU time (standard Steepest Descents), and of course the mean normalised error and the percentage correct figures became worse.

- The techniques which attempted an approximation to the inverse Hessian produced 100% correct answers. Their normalised error figures were in all cases smaller than those of the other techniques and they tended to require fewer iterations and function calls.
- There is no advantage in using an exact line search within the minimisation procedures. In each instance, the exact line search methods required about the same number of iterations, but the number of function calls per iteration was much greater (typically by a factor of three) which leads to a corresponding greater CPU time. Using an approximate line search does not degrade the normalised error, or the percentage of correct answers.
- The worst technique in terms of percentage correct is the conjugate gradients with Update3. However, the mean CPU time to obtain the answers which were correct were amongst the smallest. The following section shows a problem where Update3 gives better results than the best conjugate gradients technique in this table. Perhaps this illustrates that Update3 can be numerically unstable due to a denominator which can potentially go to zero.
- The best technique in terms of smallest normalised error is the Levenberg–Marquardt method. However, although it required fewer iterations than any other technique, the mean CPU time was longer than several other techniques due to the required matrix inversion in the algorithm.
- The fastest techniques which also had a small overall normalised error, are the quasi-Newton methods with approximate line search.

5.2 The XOR function

The XOR function may be considered to be the deterministic, two bit equivalent of the previous problem. Specifically, the total possible set of patterns consists of the four binary sequences, 00, 01, 11, 10, and the output for each pattern is either '1', if the number of '1's in the input pattern is odd and '0' otherwise. This problem has been considered interesting because patterns which are close in the input space (in terms of the Hamming distance) are maximally apart in the output space and thus a simple linear transformation method would not be capable of representing this mapping. It is not, of course, a 'typical' problem — quite the contrary. However, it is very small scale, allowing statistics to be built up easily, and the error surface has a sufficient number of unsuitable local minima (shallow flutes which propagate out to infinity) to prevent the adaptive network obtaining the correct answer every time for a random start configuration of the weights. This is in contrast to the previous problem where most of the search strategies could find a suitable minimum almost 100% of the time, and thus their relative abilities to find a good solution was not explored. In this case, the ability of the different learning strategies to obtain the *suitable minima may be examined, as well as their efficiency in arriving* at such a minimum. We have found that the extension of the XOR function to N-bit parity rapidly introduces many unsuitable minima and it is very difficult for *any* search strategy to find a single good solution for a generic starting point for $N \approx 10$.

The XOR experiment consisted of a series of runs for each one of various search strategies (see Table 2). Each run involved presenting the four input patterns, /00, 10, 11, 01/,

	% Correct	Number of iterations	Mean Error, $\mathcal{E}$	Number of function calls	Mean CPU (seconds)
S.D. (approx.)	85.1	420	0.1297	2526	7.84
C.G. (P-R & approx.)	67.4	40	0.2802	239	0.75
C.G. (F-R & approx.)	67.5	43	0.2835	241	0.78
C.G. (F-R & exact)	68.2	47	0.2765	846	2.35
C.G. (Update3 & approx.)	75.4	38	0.2161	237	0.74
q-N (BFGS & approx.)	62.8	26	0.2928	155	0.59
q-N (DFP & approx.)	47.3	56	0.4157	391	1.36
L-M (approx.)	94.0	10	0.0432	54	0.46

S.D = steepest descent, C.G = conjugate gradients,
q-N = quasi-Newton, L-M = Levenberg-Marquardt,

Table 2: Results Table illustrating the performance of various search strategies on the XOR function problem using a 2-2-1 standard layered network with linear output units.

together with the corresponding desired target patterns, /0, 1, 0, 1/, to a network with two input units, two logistically nonlinear hidden units and one linear output unit. From a random configuration of start weight values, the chosen search strategy was employed until a minimum error was achieved. This procedure was repeated for *one thousand* different sets of random start positions taken to be in the range $(-1,1)$ for the initial network weights and statistics collected as in the previous problem. The results are summarised in Table 2. Note that a 'correct' answer was assumed if the normalised error was less than 0.05. This criterion is equivalent to taking the nearest output value of either zero or unity as long as the output is not near the intermediate value of 1/2 (in which case it is assumed to be wrong).

Several points can be made by reference to Table 2.

- The slowest method is steepest descent which took an order of magnitude longer than the other techniques, requiring an order of magnitude more iterations, or search directions. However, it produced the second best set of percentage correct results.
- Interestingly, the best update strategy for conjugate gradients is not the same as in the previous problem. Indeed, Update3 has very good performance in the XOR problem, whereas it produced the worst results in the Markov Source Discriminator problem. The reason for this is unclear, but is probably indicative of numerical instability induced by a possibly small denominator in the Update3 method. For this reason, we will not consider this update method again, in spite of the good results for this particular problem. Generally, conjugate gradients produces a 'correct' answer almost 70% of the time. A more precise line search method is included to indicate that slightly better performance is possible, in terms of percentage correct and normalised error, but by taking about three times as long to do so.
- The quasi-Newton methods give slightly worse results than conjugate gradients.

- The substantially superior method is the Levenberg–Marquardt strategy with 94% correctness obtained with fewer iterations and function calls in less time and with a smaller mean normalised error than any other method.

The reason for steepest descents good performance in this problem is not evident. However, since the Levenberg–Marquardt strategy initially performs a steepest descent motion, it may be that the many unsuitable local minima near the origin in this problem, are 'smoothed out' by an initially coarse evolution. Conjugate gradients and the quasi–Newton methods are looking for a minimum immediately.

5.3 Chaotic time series prediction.

An interesting application of layered network models is that they have a limited capacity to predict, within a reasonable tolerance, time series generated by nonlinear dynamical systems. It is currently an active research area [12,13,14] to use various techniques to try to estimate the future values of an ordered sequence of numbers generated by 'chaotic' maps (deterministic generators where initial, arbitrarily small uncertainties are exponentially magnified so that subsequent behaviour is apparently random). A layered network performs this task by attempting to construct an interpolating surface which is as close as possible to the mapping surface of the given dynamical system according to some training data.

The time series considered in this subsection is the apparently very simple *quadratic map* where the value of the time series at time $t+1$, x^{t+1}, is given in terms of the time series at time t by

$$x^{t+1} = 4x^t(1-x^t) \tag{57}$$

This map is known to be chaotic on the interval $[0,1]$. In particular, the autocorrelation function of this time series may be shown to be delta distribution correlated [15], i.e. $\langle x^n x^{n+k}\rangle \propto \delta_{0,k}$. Thus, according to second order statistics, this time series would appear to be white noise; the 'memory' of the past is 'forgotten', totally, at each time step and indeed any conventional long or short term spectrum analysis would corroborate this opinion. However, the map is deterministic and so given the initial starting value exactly, the value of the time series n steps into the future may be analytically obtained (see Appendix C in [15] for further details).

Thus the time series may appear to be intrinsically extremely complicated, whereas it is completely deterministic and has been generated by a smooth, continuous and differentiable map (a quadratic functional). Consequently, there is every reason to suppose that a layered network model should be able to estimate reasonably the next iterate in the series (and indeed the next N values for some small N) given only the current value, once the network has been trained on a sufficient and representative training set.

The experiment proceeds as follows. The input training pattern is simply a data point in the interval $[0,1] \in \mathbf{R}$. The desired target pattern is the number obtained from the mapping in Equation (57). The network employed to try and represent the mapping is one input unit, six logistically nonlinear hidden units and one linear output unit all fully connected between adjacent layers. Training, involved presenting a set of 100 input–output pairs of patterns (real numbers) where the input numbers were distributed according to a uniform

	% Correct	Number of iterations	Mean Error, $\mathcal{E}$	Number of function calls	Mean CPU (seconds)
S.D. (approx.)	1	4801	0.1575	31100	4809.0
C.G. (P-R & approx.)	59	192	0.1679	1285	202.5
C.G. (F-R & approx.)	70	135	0.1013	838	133.7
q-N (BFGS &approx.)	97	143	0.0303	788	131.3
q-N (DFP &approx.)	95	110	0.0044	720	115.4
L-M (approx.)	100	58	0.0016	341	109.2

S.D = steepest descent, C.G = conjugate gradients,
q-N = quasi-Newton, L-M = Levenberg-Marquardt,

Table 3: Results Table illustrating the performance of various search strategies on the quadratic map problem for a 1–6–1 standard layered network with linear output units.

random measure on the interval $(0,1)$[7]. The various search strategies were applied to obtain a set of weight values for the network for a given random set of start weights. Once the network had been trained, it was tested on another 100 random patterns chosen over the same interval with a different random seed start. Since the actual values are known for the test set, the normalised error was computed for each search strategy. This was repeated for 100 different random start configurations of the network. Note that for steepest descents, we limited the maximum number of iterations, or search directions to be 5000 due to the excessive computation time that this strategy demands. Table 3 displays the success and efficiency of each technique in producing a sufficiently close graph in $\mathbf{R}\otimes\mathbf{R}$ to the quadratic map. Note that, since the mapping to be approximated is very smooth, the network can usually produce a very good fit which results in a very small normalised error. For this reason, our criterion for defining the 'percentage correct' solutions is to fix the upper limit on the normalised error to be $\mathcal{E} = 0.01$. This is quite a strict tolerance, but it does allow a good separation of the various search strategies, as the Table illustrates.

From Table 3, we may conclude that, as far as the quadratic time series prediction is concerned:–

- Steepest descents is the poorest strategy in terms of the percentage correct and the mean CPU time (20 times more than conjugate gradients). However, because an upper limit of 5000 iterations was imposed, it is possible that a greater percentage of correct solutions would be obtained at the expense of greater CPU time. Out of the 100 experiments, 25 had completed with an obviously incorrect solution and one experiment produced a 'correct' answer. Thus the best possible percentage correct that one could hope for in this set of experiments is 75% assuming that all the truncated experiments would have eventually converged to the correct solution.
- The best method, with 100% correct and with the smallest mean CPU time is the Levenberg-Marquardt method. It requires almost half the iterations of the quasi-Newton methods.

[7]Because this map is known to be ergodic, it is permissible to use an ensemble average over the interval, being equivalent to the sequential time series.

- Both quasi–Newton methods gave close to 100% correct. The DFP version was slightly faster than BFGS but produced slightly worse percentage correct figures. Interestingly, the DFP method gave a much smaller normalised error than BFGS even though BFGS produced more correct answers. Thus, when BFGS failed, it failed with a much larger normalised error than the DFP method. In the three instances in which it failed, BFGS produced a normalised error of approximately unity, *ie.* it was predicting in the mean.
- Conjugate gradients strategies produced worse percentage correct figures than quasi-Newton methods, but without taking many more iterations.

5.4 Point Source Location Using Focal Plane Arrays.

This example represents an intermediate stage between the small–scale problems of the previous examples and the large–parameter speech recognition experiment discussed in the following section. In this example, the number of parameters is much larger than the Markov Source Discriminator problem, but not so large that storage requirements preclude the use of the BFGS and Levenberg–Marquardt methods. Therefore, a comparison of all the methods can be made.

The problem addressed in this section is one of multi–sensor data analysis. Data generated by a given sensor system represents a particular view of the scene under consideration. The signal processing problem is to provide a description of that scene, given some *a priori* knowledge of the scene characteristics and knowledge of the properties of the sensor. For example, in an active radar situation, the scene may include point scatterers, distributed scatterers, clutter, chaff and interference. One common form of *a priori* knowledge imposed is the special case that the scene can be represented spatially by a collection of point sources. There are many techniques for estimating the parameters of sources in a scene, all of which require some form of training of the system [16,17]. This may be achieved by moving a single source around in the far field and recording the output of the system. If the outputs of all sensors in the system are sampled simultaneously, then we obtain a vector of numbers, the 'image vector', which gives a snapshot from the system for a given source position. All these vectors are collected together as columns of a matrix which forms a 'reference library' of signals expected from each incident direction. This library is termed the array manifold [17,18].

The specific problem we shall consider here is the estimation of the position of a single source in the scene given its sampled image vector. The library of vectors is generated from the outputs of a 4×4 array of elements in the focal plane of an imaging system, giving an image vector of dimension 16 as follows.

The two–dimensional imaging equation relating a time–varying image $g(x, y; t)$ to the scene, $f(\xi, \eta; t)$ is given by a convolution equation of the form

$$g(x, y; t) = \int\int h(x, y; \xi, \eta) f(\xi, \eta; t) d\xi d\eta + n(x, y; t) \tag{58}$$

where $h(x, y; \xi, \eta)$ is the point–spread function of the imaging system and $n(x, y; t)$ is the noise in the degraded image. When the image is sampled, the image is known at only a finite number of points in the image plane $(\mathbf{r}_1, \ldots \mathbf{r}_{16})$ corresponding to array element positions

$\mathbf{r}_i = (x_i, y_i)$ and Equation (58) becomes

$$g_i(t) = \int\int h_i(\xi,\eta) f(\xi,\eta;t) d\xi d\eta + n_i(t) \tag{59}$$

where $g_i(t)$ *is the value of the sampled image at position* $\mathbf{r}_i$ *at time* t. The function $h_i(\xi,\eta)$ is the response at the position $\mathbf{r}_i$ to a point source in the far-field as a function of position of that source. For an ideal, diffraction-limited, space-invariant imaging system (one which acts uniformly across image and object planes) with a circular aperture, this response is given by [19]

$$h_i(\xi,\eta) = \frac{\Omega J_1(\Omega r_i)}{2\pi r_i} \tag{60}$$

where Ω is the spatial bandwidth, $r_i = [(\xi - x_i)^2 + (\eta - y_i)^2]^{\frac{1}{2}}$ and $J_1(x)$ is a Bessel function of the first kind of order one.

In this example, we take $\Omega = \pi$, so that sampling at the Nyquist rate, (π/Ω) gives unit spacing *of the sample points (see Figure 2). Noise is not considered* and so a *point source* in the far field at a position ξ_0, η_0 gives rise to an image vector

$$\mathbf{h}(\xi_0,\eta_0) \equiv (h_1(\xi_0,\eta_0), h_2(\xi_0,\eta_0), \ldots, h_{16}(\xi_0,\eta_0)).$$

Training consists of presenting images of a point source at different positions in the far field as input with the corresponding (ξ,η) positions of the source as targets. The image vectors are normalised and the input data comprises the normalised images of the source at positions which lie on a 21×21 grid covering the field of view of the array (there are 6 image positions per sample spacing). Figure 2 shows the positions of the 16 sample points and, in the same coordinate system, the positions of the sources in the far field used for training and testing the network. The target data used for training are the (ξ,η) positions on the grid of the source which gives rise to the image. Thus, for the training phase, there are 441 patterns of dimension 16 for the inputs and dimension 2 for the targets. For the test data, the normalised images of a single source at 400 positions which are randomly distributed across the field of view are taken as input with the positions as targets.

The focal-plane array illustration described is highly idealised. In general, the array manifold, and the image vectors, would be complex vectors and some method of incorporating complex vectors into a feed-forward network would have to be considered. This is not a difficult task, but for our purposes we shall restrict the example to considering real vector inputs only.

The network chosen to represent the mapping has 12 logistically nonlinear hidden units and linear output units, giving 230 weights for which to solve in the nonlinear minimisation routines. For the point-source location problem, the criterion chosen as to whether a 'correct' solution was obtained was that the normalised error should be less than 0.015. This corresponds to the square root of the mean squared error in the position at the output of the network being less than one quarter of a grid spacing. For a normalised error less than this, we can assume that the network has encoded the mapping and is interpolating the training data to find positions corresponding to the test images. For a normalised error much greater than this, the network has failed to perform interpolation of the training data. The experiment was run for 10 different random start configurations for the weights and the results are shown in Table 4. The experiments for steepest descents were truncated after 5000 iterations.

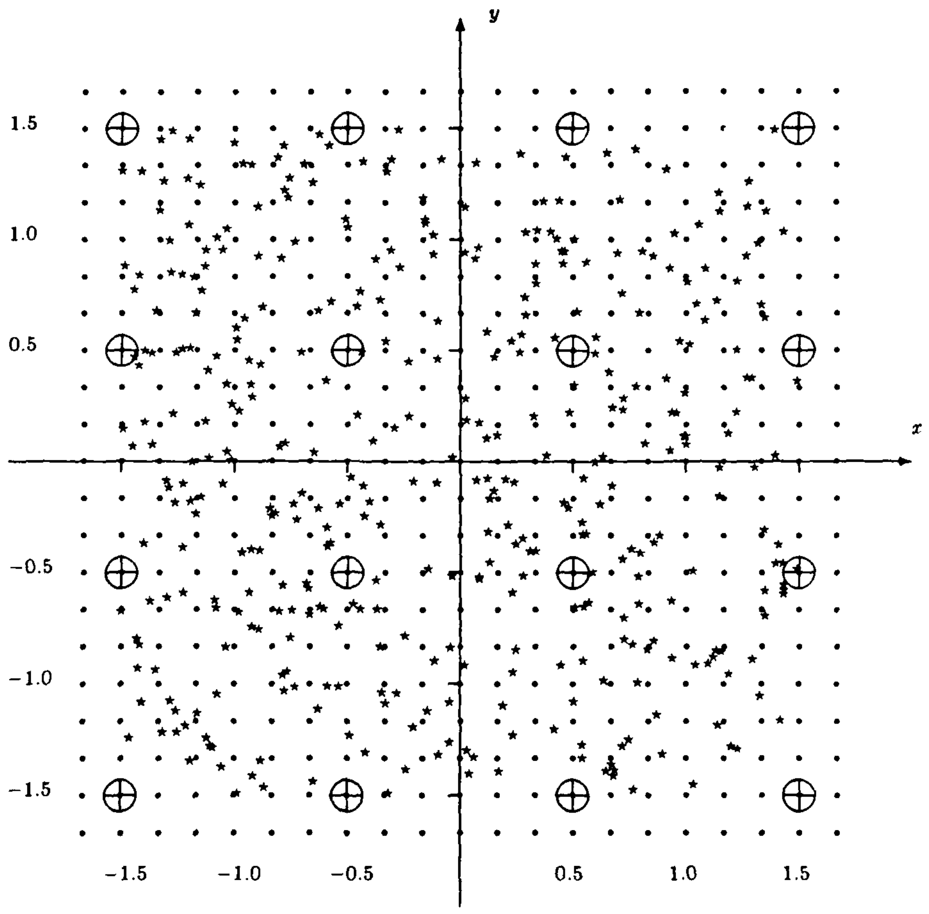

Figure 2: Diagram showing the calibration target positions for the training data (circles), the target positions for the test data (stars) and the sensor positions (circled crosses) plotted in dimensionless units.

	% Correct	Number of iterations	Mean Error, $\mathcal{E}$	Number of function calls	Mean CPU (seconds)
S.D. (approx.)	0	> 5001	0.0385	≫ 30000	78509
C.G. (P-R & approx.)	100	1582	0.0058	9030	23260
C.G. (F-R & approx.)	100	1930	0.0051	11076	28414
q-N (BFGS &approx.)	100	301	0.0031	1263	4312
q-N (DFP &approx.)	100	301	0.0034	1385	4468
L-M (approx.)	100	110	0.0016	604	37493

S.D = steepest descent, C.G = conjugate gradients,
q-N = quasi-Newton, L-M = Levenberg-Marquardt,

Table 4: Results Table showing the performance of the search strategies on the synthetic Focal Plane Array problem using a 16-12-2 standard layered network with linear output units (230 weights to adjust).

From Table 4 we may infer:-

- The weakest strategy is steepest descents. Because of the imposed cutoff on the maximum number of allowed iterations, this method did not produce any 'correct' answers. Consequently it is not possible to give the number of iterations and the number of function calls per correct solution. However, had these experiments eventually converged, then more than 5000 iterations and more than 30000 function calls would have been required for each correct solution. Also each experiment would have required in excess of one days CPU time.
- In contrast, all other techniques produced 100% correct solutions. In terms of function and gradient calls, the Levenberg-Marquardt method was superior and conjugate gradients was worst. However, because of the critical size of the network (the total number of adjustable parameters – 230) the total time required *per iteration* was much greater for the Levenberg-Marquardt method than the other successful procedures. In particular, even though the Levenberg-Marquardt method required an order of magnitude fewer function calls and iterations than conjugate gradients, it actually took 60% longer on average for each experiment.
- In terms of speed, the quasi-Newton methods are superior for this problem with a smaller normalised error compared with conjugate gradients. There is no real distinction to be made between either DFP or BFGS.

5.5 Speech recognition: the 'EE' set.

This final problem is more typical of large scale problems. It is a speaker-dependent speech recognition experiment where the words to be classified are the highly confusable eight 'EE' sounds from the alphabet, i.e. { 'B', 'C', 'D', 'E', 'G', 'P', 'T', 'V' }. Speech is best represented as a local frequency pattern which evolves in time (e.g. a spectrogram) and thus the pattern representing an isolated word may be visualised as a static distribution of energy density in time and frequency. The particular time-frequency representation used in this experiment was the output of the JSRU hardware channel vocoder [20]. This performs a filter bank analysis of the speech waveform which produces an average spectral energy density in each of nineteen frequency channels, nonuniformly distributed across the approximate range of 200Hz $\sim$ 4kHz. Such a frequency slice is produced fifty times a second, and so an isolated word of typical duration $0.5 - 1$ second will require 25–50 time slices. The isolated words considered in this experiment were contained within 40 time slices and hence each word corresponds to an array of numbers where the array is of a size $40 \times 19 = 760$. In addition, the allowable output intensities in each time-frequency slot is quantised to 15 levels taken to be real numbers in the range $[0, 1]$.

A single consistent speaker produced 40 utterances of each of the eight desired target sounds which were converted into the time-frequency patterns as described above. Twenty sets of these eight patterns were used to train the layered network model (i.e. 160 patterns in total) and the remaining utterances were used to test the ability of the trained network to generalise. The desired output target patterns were taken to be a one-from-eight coding (a '1' in the place corresponding to the input training patterns word and a '0' elsewhere). Thus the number of input units to the network was 760, the number of hidden units was taken to be 8 and the number of output units was also 8. Again, linear output units were employed and the 'logistic' nonlinearity was used on each hidden unit.

A decision had to be made when the network had produced the 'correct' answer for a given input pattern. Of course, an upper limit on the normalised error could be employed, but in this instance it is more sensible to choose those instances when the actual output vector of the network is closest to the desired target pattern. Table 5 shows the percentage correct on the test data, the normalised error and the efficiency (in terms of iterations and time required) of obtaining the correct answers using steepest descents and conjugate gradients.

Note that the scale of the problem (as determined by the number of weights between the input and hidden units > 6000) means that the techniques which require storage, update and possibly the inverse of an $N \times N$ matrix are infeasible. Thus, we were unable to test the quasi-Newton and Levenberg-Marquardt methods on this speech problem. This is an instance where techniques do not scale favourably with the size of a problem, which may be important in typical speech experiments. However, the limitation is not one of CPU time diverging faster than N (as may be the case for K-nearest neighbour classification for instance) but storage requirements diverging as N^2 which would be surmountable if an N-storage update scheme could be devised for these methods. This is unfortunate, but is characteristic of the pitfalls to be aware of when extrapolating techniques and knowledge obtained on small scale problems to large scale, real world data. Of course, speech should not be coded as a static pattern of real numbers in a two dimensional array. The metric employed does not differentiate between the ordering of the points in an array, and so this scheme does not allow for the causal flow of information relevant to speech. *Time* and Frequency are complementary concepts and each frequency slice at each time slot should be treated allowing for the time ordering of each slice. Schemes which do this will not deal with a static pattern of 760 numbers, but a sequence of 19 ordered r~al numbers, thus reducing the scale of the problem.

Note that the size of the network used for this problem, and the time available precluded

	Number Correct	Number of iterations	Error, $\mathcal{E}$	Number of function calls	CPU Time dd:hh:mm
S.D. (approx.)	153	6663	0.5422	28520	4:10:56
C.G.[1] (P-R)	128	441	0.5205	1946	0:07:10
C.G.[2] (P-R)	137	465	0.4756	1924	0:07:22
C.G.[3] (P-R)	144	2726	0.4226	12340	1:21:38
C.G.[4] (P-R)	150	2180	0.3764	10140	1:13:24
C.G.[5] (P-R)	143	484	0.5039	2013	0:07:43

S.D = steepest descent, C.G = conjugate gradients (five experiments),

Table 5: Results Table illustrating the performance of various search strategies on the 'EE' set speech recognition data. The network was a 760–8–8 standard layered network with linear output units.

an extensive set of experiments. In particular, the steepest descents procedure was only used on one experiment and conjugate gradients on five experiments. In addition, steepest descents was used to train the network incrementally. That is, forty examples of the total one hundred and sixty training patterns were used to obtain an initial set of 'representative' weights which were then used as initial values to train the network on the whole training set using steepest descents. Thus, the Table gives the number of individual patterns which were classified 'correctly' according to whether the output vector of the network was closest in a Euclidean distance sense to the appropriate target vector. The results are presented for each experiment performed.

- On the limited data available, steepest descents gave the best classification accuracy although the normalised error was the worst. However, this result was obtained from a 'good' starting position by incrementally training on a subset of the data and it still required in excess of four days to find a minimum.
- Experiments [3] and [4] in conjugate gradients which required over a day CPU time, both obtained a very small normalised error on training (less than 0.005). This compares with normalised errors of 0.379 for the other three experiments on the training set. Since the performance on the test set is comparable, this is indicative of the fact that a good solution (in terms of a small error on the training set) is not necessarily a criterion for producing good results (in terms of classification) on a previously unseen test set.

6 Discussion.

This memorandum has been concerned with the suitability of a range of nonlinear optimisation strategies applied to the minimisation of the error produced by an adaptive feed-forward layered network.

Before we state the conclusions, it will be productive to point out certain constraints in the approach we have followed. In order to restrict the total number of experiments, several assumptions have been made and several parameters have been prescribed. In all the experiments, the parameters of the network, the 'weights', have been generated randomly from a uniform distribution on the range $(-1, 1)$. Our experience has shown that this is a reasonable range for the starting values of a network for a wide range of problems provided the input and output data set is suitably scaled. Nevertheless, a different range for the starting values may result in a different set of solutions for the minima of the error function. Certainly, the distribution of the minima obtained by the algorithms, and hence the number of 'correct' solutions would be different.

A further constraint has been the choice of convergence criterion for the algorithms and, in particular, the value of the convergence factor, *ftol*. In some examples, it may be possible to tolerate a larger value for *ftol*, giving a reduced time for solution at the cost of only a marginal deterioration in performance. The value adopted reflects a compromise between length of computation time and premature termination. Alternatively, the termination condition of the algorithms could have been chosen so that the experiments completed after a fixed number of iterations or when the error fell below some prescribed threshold.

The choice of functions for the nonlinearities of the hidden units and the output units should reflect, in practice, *a priori* knowledge about the classification problem under consideration. However, in all the examples presented in this memorandum, the hidden units have been chosen to have a logistic nonlinearity, and the output units have a linear transfer function.

One important feature of the application of feed-forward layered networks to pattern classification problems, and one which our approach has in common with many conventional approaches, is the use of a set of patterns for training the network and a set for testing the network. Whilst this may be a satisfactory strategy for many applications, often the use of a third data set, an 'evaluation' set, is required. Minimising the error at the output of the network is not equivalent to maximising classification accuracy, and an evaluation set would be used to assess the solutions produced by training and select one of them for application to an unseen test set.

To train the network, we have assumed that the error function and its derivatives with respect to all the adjustable parameters of the network can be evaluated. For a network of a given size, the computation required to evaluate the derivatives increases linearly with the number of patterns in the training set. For problems with a large number of training patterns, the computation involved in calculating the set of derivatives may be prohibitive. In such a situation, it may be advantageous to use 'representative' subsets of the training patterns in the calculation of the derivatives. The parameters of the network may then be adjusted after each representative set is presented to the network. Although this strategy may not result in a local minimum of the error function being attained (since the error surface for each subset of data will be different in general), it does allow a more rapid update of the parameters, and produces a set of values which may be close to the locally optimum set.

On the basis of our studies on problems ranging from the small-scale (networks with only a few adjustable parameters) to the large-scale (networks with several thousand adjustable parameters) we may infer several points.

First, there is no advantage in pursuing an accurate line search method within the nonlinear optimisation, because each experiment requires much more CPU time with only marginal improvement, if at all, in the classification accuracy. The usefulness of an accurate line search is confined to the insight it produces in deriving theorems regarding *quadratic termination*.

Depending on the size of the network under consideration, different optimisation schemes emerge as the most suitable. For small-scale problems where the network typically involves $\approx$ 30 adjustable parameters, the best method is the Levenberg-Marquardt approach (see the XOR function and the chaotic time series prediction problems). Medium-scale networks with between 40 and 250 parameters seem to be most efficiently solved by a quasi-Newton method with, perhaps, the BFGS variant showing an overall slight advantage. For large-scale networks with much more than 250 parameters, the best methods cannot be applied due to the requirement of N^2 storage. Thus our choice for large-scale problems is the conjugate gradients technique, implemented with the Polak-Ribiere update strategy.

Steepest descents is *not recommended for any real problem due to its inefficiency*. However, there are instances where steepest descents is a good method to use. The problems with steepest descents lie in the arbitrary choice of a step length (in the absence of a line search) and a momentum term which is extremely sensitive to the particular problem and network geometry under consideration. To obtain even a 'reasonable' set of constants may take many heuristic experiments. Because steepest descents is so slow to evolve to a local *minimum, the method is not often run to convergence. Instead*, an arbitrary *cutoff is im-*posed on the maximum number of iterations. It turns out that this is a good strategy for certain problems where it may not be desirable to obtain a good nonlinear optimisation of a least-mean-squares error criterion. In particular, if there are not as many equations to solve produced by the training set, as adjustable parameters in the network, it may be possible to obtain an infinity of solutions (sets of values of the adjustable parameters) which all give zero error. This is certainly true in a totally linear network where this situation corresponds to specifying an underdetermined problem. What this implies is that, for the given network geometry, it is possible to fit exactly all the training data in spite of the fact that the data will be noise corrupted. Thus, if the aim is **not** to model the noise and the data together, it is a poor philosophy to try and produce an interpolating surface which passes exactly through all the known data points. In such situations the capability of the *network to generalise to previously unseen data is diminished. Minimising the error on* a training set is not equivalent to maximising the classification accuracy on a test set.

For underdetermined problems it is desirable to try and compensate for the redundancy by imposing extra constraints on the adjustable parameters, or by employing methods which artificially increase the training set. Alternatively, one could adopt the view that the output error of the network for an underdetermined problem should **not** be minimised. In which case there may be no real advantage in employing an efficient nonlinear optimisation technique. For instance, one can observe that standard steepest descents is capable of performing better (in terms of generalisation accuracy) since the imposed step length, momentum rate and maximum number of iterations effectively prevent a good local minimum to be found in the high dimensional error surface. Note that the 'EE' set example is such a problem [there are 6160 parameters, but only 160 training patterns which, with 8 output units, gives 1280 equations to solve] and it is evident from Table 5 that the steepest descent experiment produces slightly better generalisation accuracy compared to the conjugate gradients experiments.

An alternative situation occurs, when there are many more training patterns than adjustable parameters. This corresponds to an overdetermined problem for which, in the linear case, there is no unique solution generally. All one can do in this case is to obtain a set of parameter values which produce the smallest error (one could also impose further requirments to obtain a unique solution with this smallest error). The (least squares) solution obtained produces a surface which passes close to, but does not pass directly through, all the known training data points. An approximate interpolation surface is generated. This is reminiscent of band limited interpolation and reconstruction of one dimensional signals. An effective band limit is imposed by the geometry of the adaptive layered network which restricts the degree of irregularity that the network is capable of modelling. Thus, for an inherently 'rough' data set such as a set of structured patterns corrupted by noise, the network produces a 'best fit' interpolation scheme which is smoothing out the noise to produce a model of the underlying structure in the data. If this implicit nonlinear interpolation scheme is smooth enough to suppress the noise, and yet complex enough to embody the relationships in the data, then the network would be expected to generalise to previously unseen data with the same structure. This is the situation discussed in the chaotic time series prediction experiment, and the synthetic radar problem. It is within this general scenario of an overdetermined problem where the adaptive layered networks are most productive and where it is advantageous to use an efficient search strategy as Tables 1 and 4 illustrate.

This memorandum has considered, exclusively, nonlinear optimisation of the function produced at the output of a layered network structure. However, the precise functional rôles of each layer of weights or adjustable parameters is different. In particular, in the case of a 1-from-n classification task where the output nonlinearities are at most invertible, the hidden–output weights perform a *linear* least mean squares classification which maximises a measure connecting the within class and the between class covariance matrices at the outputs of the hidden layer. Thus, there are situations where the obvious strategy is to adjust the weights connecting the input and hidden layers by a nonlinear optimisation scheme, but at the same time adjust the weights connecting the hidden and output layers by a linear optimisation method. Whether this is an overall better procedure has yet to be conclusively demonstrated.

Other areas in which this work may be generalised are as follows.

1. Study the scaling behaviour of the random start weights in relation to the magnitudes of the input and output training patterns and the relationship with convergence of the nonlinear optimisation schemes.

2. Try and devise an optimisation scheme which embodies the philosophy of the quasi–Newton approaches, but only requires N–storage to tackle large-scale problems.

3. Study the effect of 'bootstrap training' — the evolution in the error surface produced by first presenting a subset of the training patterns, then a larger subset and so on. Thus, only partial, or approximate, knowledge of the gradient is allowed at any instant. What effects will this incomplete knowledge have on the optimisation schemes which assume accurate information of the function to be minimised and its first derivative?

4. Study the effects of imposing possible constraints on the possible parameter values (*such as boundedness, minimum norm solutions, finite word length accuracy, etc.*).

5. Study the general problem of the robustness of the solutions obtained by nonlinear optimisation schemes as the network is degraded by some means, ie. do some techniques produce solutions which exhibit 'graceful degradation' as the network structure is progressively corrupted in some manner.

However, these are topics for the future.

References

[1] William H. Press, Brian P. Flannery, Saul A. Teukolsky and William T. Vetterling. *"Numerical Recipes: The Art of Scientific Computing"*, (Cambridge University Press, 1986).

[2] J.C. Nash, *"Compact numerical methods for computers: linear algebra and function minimisation"*, (Adam Hilger Ltd., Bristol, 1979).

[3] *"Numerical Solutions of Systems of Nonlinear Algebraic Equations"*, edited by George D. Byrne and Charles A. Hall, (Academic Press, New York and London, 1973).

[4] H.Y. Huang, *"Unified approach to quadratically convergent algorithms for function minimisation"*, J. Optimisation Theory Appl., **6**, 269–282, (1970).

[5] L.C.W. Dixon, *"The choice of step length, a crucial factor in the performance of variable metric algorithms"*, in *Numerical Methods for Nonlinear Optimisation*, ed. F.A. Lootsma, Academic Press, London, (1972).

[6] C.G. Broyden, *"Quasi–Newton methods and their application to function minimisation"*, Math. Comp., **21**, 368–381, (1967).

[7] Levenberg, K., *"A Method for the Solution of Certain Non-linear Problems in Least Squares"*, *Quart. Appl. Math.*, **2**, 164-168, (1944).

[8] Marquardt, D.W., *"An algorithm for Least-squares Estimation of Nonlinear Parameters"*, *J. Soc. Indust. Appl. Math.*, **2**, No. 2, 431-441, (June 1963).

[9] Dennis, J.E.: *"Some Computational Techniques for the Nonlinear Least Squares Problem"*, in Byrne, G.D. and Hall, C.A. (eds), *"Numerical Solution of Systems of Nonlinear Algebraic Equations"*, (Academic Press, 1973).

[10] Marvin L. Minsky and Seymour A. Papert, *"Perceptrons: An Introduction to Computational Geometry"*, (MIT Press, expanded edition, 1988).

[11] Mark D. Bedworth and John S. Bridle, *"Experiments with the Back-Propagation Algorithm: A Systematic Look at a Small Problem"*, (RSRE Memorandum 4049, June 1987).

[12] Alan Lapedes and Robert Farber, *"Nonlinear Signal Processing using Neural Networks: Prediction and System Modelling"*, (submitted to Proceedings of IEEE, July 1987).

[13] J. Doyne Farmer and John J. Sidorowich *"Predicting Chaotic Time Series"*, *Physical Review Letters*, **59**,(8), 845–848, (1987).

[14] Martin Casdagli, *"Nonlinear prediction of chaotic time series"*, (Submitted to Physica D 1988)

[15] D.S. Broomhead and David Lowe, *"Radial Basis Functions, Multi-Variable Functional Interpolation and Adaptive Networks"*, *R.S.R.E. Memorandum 4148*, (March 1988).

[16] Clarke, I.J.: "Comparison of Advanced Signal Processing Algorithms", Eurocon'86, Paris.

[17] Mather, J.L.: "Least Squares Solutions in Signal Processing Using the Singular Value Decomposition", RSRE Memo. 3864, January 1986.

[18] Schmidt, R.O.: "A Signal Subspace Approach to Multiple Emitter Location and Spectral Estimation", PhD Thesis, Stanford University, November 1981.

[19] Andrews, H.C. and Hunt, B.R., "Digital Image Restoration", (Prentice-Hall, 1977)

[20] J.N. Holmes, *"The JSRU channel vocoder"*, *IEE PROC.*, **127**, 53-60, (1980).

[illegible]

DOCUMENT CONTROL SHEET

Overall security classification of sheet .....UNCLASSIFIED...

(As far as possible this sheet should contain only unclassified information. If it is necessary to enter classified information, the box concerned must be marked to indicate the classification eg (R) (C) or (S))

1. DRIC Reference (if known)	2. Originator's Reference MEMO 4157	3. Agency Reference	4. Report Security Classification U/C
5. Originator's Code (if known) 778400	6. Originator (Corporate Author) Name and Location Royal Signals and Radar Establishment St Andrews Road, Great Malvern, Worcs. WR14 3PS		
5a. Sponsoring Agency's Code (if known)	6a. Sponsoring Agency (Contract Authority) Name and Location		

7. Title
A COMPARISON OF NONLINEAR OPTIMISATION STRATEGIES FOR FEED-FORWARD ADAPTIVE LAYERED NETWORKS.[a]

7a. Title in Foreign Language (in the case of translations)

7b. Presented at (for conference papers) Title, place and date of conference

8. Author 1 Surname, initials Webb, A.R.	9(a) Author 2 Lowe, D.	9(b) Authors 3,4... Bedworth, M.D.	10. Date 1988.7	pp. ref. 33
11. Contract Number	12. Period	13. Project	14. Other Reference	

15. Distribution statement
Unlimited

Descriptors (or keywords)

continue on separate piece of paper

Abstract This work discusses various learning strategies which may be employed for the generic class of layered feed-forward adaptive networks exemplified by the traditional *Multilayer Perceptron*. Such a network is only useful if a set of weight values exists which allows the network to form a good approximation to an underlying (an possibly unknown) transformation between input and output patterns. This memorandum is motivated by the need for schemes which are capable of producing such a set of weights, (should one exist) as efficiently as possible. As these issues are dependent on specific problem features, this memorandum considers the application of various 'learning' algorithms to examples ranging from small scale 'trivial' problems (solution of the XOR function), to larger scale pattern processing applications (speech recognition of isolated confusable whole words).

[a]Work performed at RSRE between November 1987 and May 1988.

S80/48